Remarks

Applicants respectfully request reconsideration. A Notice of Appeal and Pre-Appeal Request for Review for the above-identified patent application were filed on September 4, 2007. The Pre-Appeal Board returned its decision on November 23, 2007, indicating that the application was to proceed to the Board of Patent Appeals and Interferences.

This amendment is intended to end the appeal and reopen prosecution on the merits. To that effect, this amendment is accompanied by a Request for Continued Examination (RCE) and the requisite fee.

Applicants' attorney wishes to extend his thanks to Examiner Mais for conducting a telephonic interview on December 20, 2007. Independent claims 1, 25, and 26 were discussed with reference to the prior art of *Baker* et al. (US Patent No. 6,266,700, hereinafter, "*Baker*"). Although no specific agreement was reached on any proposed claim language, the Examiner indicated that his interpretations of the terms "instantiating" and "instance," which are found in each of the claims 1, 25, and 26, were more generic than Applicant had intended, and that language more targeted to the Applicants' use of these concepts might distinguish over *Baker*. The Examiner also stated that *Baker* does not appear to disclose a "software program," as recited in each of claims 1, 25, and 26, but that this fact was not enough to avoid a rejection of Applicants' method claims based on *Baker*, since the Examiner did not give this structural limitation patentable weight in the context of the method claims. The Examiner further agreed to conduct an interview with Applicants' attorney after the filing of the RCE accompanying this amendment and before the first office action on the merits.

Turning now to the substance of this amendment, it is seen that changes have been made to the claims and remarks have been submitted. The status of the claims is as follows:

- Claims 1, 17, 21, and 26-28 are currently amended;
- Claims 2, 4, 11, and 25 have been canceled;
- Claims 29-38 have been added;
- Claims 1, 3, 5-10, 12-24, and 26-38 are pending.

The Examiner has rejected claim 1 under 35 U.S.C. § 102(b) as being anticipated by *Baker*. Applicants have amended claim 1 and respectfully submit that claim 1 as amended distinguishes over *Baker*.

Claim 1 as amended is directed to "a method of testing equipment operatively connected to a target medium having a protocol" and recites, *inter alia*, a step of —

> providing a plurality of communication element types hierarchically representing different communication elements of the protocol, each communication element type

> being a user-defined instantiable software data type pertaining to a particular layer of the protocol [.]

The term "user-defined instantiable data type" is different from the previously recited term, "user-defined data structure." Exemplary support for this term can be found at ¶¶46 – 48 of the specification. A brief explanation of "data types" may help to clarify the difference between these terms.

As is known to those skilled in the art of software design, a "data type" is a construct used to define characteristics of data used and manipulated in a software program. Data types may be fixed or user-defined. Familiar examples of fixed data types include "int," for defining variables whose values are integers, "float," for defining variables whose values are real numbers, and "char," for defining variables whose values are text strings. In addition to these standard data types, some form of which is provided by virtually all computer languages, many languages also support user-defined data types. The specification describes an example of a user-defined data type at ¶47:

> For instance, the "C" computer language provides a "typedef" instruction for creating specific data types. Instances of a data type may be created in a computer program by declaring variables of that type. The computer program can then access and manipulate the instances of that type at runtime.

Three steps may be involved in the use of typedef. First, a user defines the data type itself by providing a typedef instruction. A simple example of a typedef instruction is "typedef int t," where a data type named "t" is defined as an integer type. Second, a variable of the user-defined type is declared within a computer program. For example, a user can include the code "t myInt" in the user's computer program. This code declares a variable named "myInt" that has type "t," i.e., the integer type in this case. Third, when the computer program is run, the variable "myInt" is accessed and manipulated. An easily accessible description of the typedef instruction can be found online at the following link:

> http://publications.gbdirect.co.uk/c_book/chapter8/typedef.html.

A copy of the page located at this link is included with this response.

Communication element types as recited in claim 1 as amended are closely analogous to user-defined data types created by typedef. As the specification states at ¶48,

> Communication element types of a bus file work much the same way. Users specify these types and can create instances of them to be used and manipulated in the context of a program.

Returning now to the rejection at hand, *Baker* does not disclose a step of "providing a plurality of communication element types" as recited in claim 1 as amended. The Examiner has

alleged that *Baker's* "programmably configurable protocol descriptions" are analogous to Applicants' "communication element types." See item 4 of the Office Action. Applicants respectfully disagree.

Baker provides a description of the programmably configurable protocol descriptions at col. 3, lines 12-16:

> The protocol descriptions may take the form of one or more protocol description files for each supported network protocol and may include a protocol header record and plurality of field sub-records having data corresponding to an associated protocol and fields defined therein.

Baker also shows an example of the protocol descriptions at Fig. 3. Col. 5, lines 4-6. Fig. 3 shows a single protocol control record and a number of field subrecords. See also col. 7, lines 21-27. Fig. 4 shows an example of a protocol control record for Ethernet. Col. 5, lines 7-10. Fig. 4a shows the various field subrecords. Col. 5, lines 11-13.

Clearly, *Baker*'s "programmably configurable protocol descriptions" have nothing to do with *data types*. *Baker*'s "programmably configurable protocol descriptions" simply provide *data*. There is a great difference between *data types* and *data*: one is a construct for organizing information whereas the other is simply the information itself.

Therefore, *Baker's* "programmably configurable protocol descriptions" do not meet the definition of "communication element types" as recited in claim 1 as amended. Therefore, *Baker* cannot perform the step of "providing a plurality of communication element types" as recited in claim 1 as amended.

Claim 1 as amended further distinguishes over *Baker* in its recitation of an "instantiating" step. After setting forth a step of "providing a software program," claim 1 as amended recites, *inter alia*,

> instantiating, by the software program, one of the plurality of communication element types to create a transmit message instance within the software program, the transmit message instance being a specific expression of the respective communication element type [.]

Baker does not disclose this step, or any similar step. Although *Baker* appears to disclose various processes for accessing the programmably configurable protocol descriptions, e.g., for filtering and parsing data, the same programmably configurable protocol description is accessed every time. Nowhere in *Baker* is an instance created in a software program, which is "a specific expression of the respective communication element type." There is simply no type-instance distinction in *Baker*.

Therefore, Baker does not disclose a step of "instantiating, by the software program, one of the plurality of communication element types to create a transmit message instance within the software program," as required by claim 1 as amended.

Baker's programmably configurable protocol descriptions are similar to constants. They are fixed values that can be accessed for performing various functions. By contrast, the "transmit message instance" recited in claim 1 as amended is not fixed. It can be manipulated by the software program. This characteristic is clearly set forth in claim 1 as amended, wherein a step is recited of "manipulating the transmit message instance within the software program." The specification supports an example of this limitation at ¶48:

> Users specify these types and can create instances of them to be used and manipulated in the context of a program. […] The program can then manipulate the instance … by establishing its settings, manipulating its data, etc.

Nowhere does *Baker* recite anything akin to this "manipulating" step. Values stored in Baker's programmably configurable protocol descriptions are fixed. No software program manipulates them.

It does appear, however, that *Baker* provides a way of changing his programmably configurable protocol descriptions for use with *different protocols*. One might be tempted to view this as "manipulating" different "instances" of a generic structure, i.e., a table of values. However, *Baker* discloses this feature only as a way of adapting to *different protocols*. In contrast with *Baker*, claim 1 as amended recites that instances of communication element types are created and manipulated for use with "a target medium having a protocol." Claim 1 as amended does not touch on events that might take place when switching protocols. *Baker* may change out the "constants" for use with different protocols, but he does not create or manipulate specific "instances" of any protocol description when operating with a single protocol.

Therefore, *Baker* fails to disclose a step of "manipulating the transmit message instance within the software program" as required by claim 1 as amended.

Claim 1 as amended further recites the steps of "instantiating … to create an expect message instance" and "manipulating the expect message instance." *Baker* does not disclose the "instantiating" and "manipulating" steps relating to expect message instances for reasons already set forth above in connection with transmit message instances.

In view of the many distinctions described above between claim 1 as amended and *Baker*, and for at least these reasons, claim 1 as amended is not anticipated by *Baker*. Therefore, the rejection of claim 1 as amended under 35 U.S.C. § 102(b) is overcome. Applicants respectfully submit that claim 1 as amended is allowable.

Claims 3, 5-10, and 12-24 depend from claim 1 as amended and are allowable for the same reasons. Claim 17 has been amended to accord with the language of claim 1 as amended.

The Examiner has rejected claims 25 and 26 under 35 U.S.C. § 102(b) as being anticipated by *Baker*. Applicants have canceled claim 25. Applicants have amended claim 26 and assert that claim 26 as amended is allowable.

Claim 26 is directed to "a method of communicating over a target medium having a protocol that supports the use of messages and words." The claim recites "providing a plurality of message types and a plurality of word types for representing communications using the protocol, each of the plurality of message types and each of the plurality of word types being a *user-definable data type represented in software*" [emphasis added].

As an initial matter, *Baker* does not disclose user-definable *data types* for representing a protocol, as recited in claim 26 as amended. As indicated above, *Baker*'s programmably configurable protocol descriptions are not data types but simply data.

Also, *Baker* does not disclose "providing a plurality of message types and a plurality of word types." *Baker*'s programmably configurable protocol descriptions each contain a protocol control record and a number of field subrecords (see Figs. 3, 4a, and 4b). These may or may not have any relevance to messages or words exchanged using a particular protocol, but they are not the same thing as message types and word types as recited in claim 26 as amended. *Baker*'s protocol control record pertains to a protocol as a whole (see Fig. 4), and the field subrecords pertain to individual fields (see Fig. 4a). None of these pertain to message types or word types as recited in claim 26 as amended. Message types and word types are simply not represented as distinct constructs in *Baker*'s model.

Therefore, Baker does not disclose "providing a plurality of message types and a plurality of word types for representing communications using the protocol, each of the plurality of message types and each of the plurality of word types being a user-definable data type represented in software," as recited in claim 26 as amended.

Further, claim 26 as amended recites a step of "arranging the plurality of message types and the plurality of word types hierarchically, with at least one message type including a reference to at least one word type." *Baker* does not disclose message types or word types. Therefore, *Baker* cannot disclose arranging these different types hierarchically.

A significant aspect of this hierarchical arrangement is revealed when message types are instantiated. Claim 26 as amended recites, *inter alia*, a step of—

> instantiating the at least one message type by the software program to create at least one message instance, each message instance being a specific expression of the

> respective message type and including an instance of each word type included by reference in the respective message type, each included instance of a word type being a specific expression of the respective word type [,]

The specification provides an example of this "instantiating" step at ¶49:

> Since message types are hierarchical constructs that include in their definitions constituent word types and field types, it is evident that "message instances" include not only instances of the messages themselves, but also instances of their constituent word types and field types.

Baker does not disclose the "instantiating" step as recited in claim 26 as amended. As described in connection with claim 1 as amended above, *Baker* does not draw a distinction between "types" and "instances." *Baker*'s programmably configurable protocol descriptions are not types, and no instances are created.

In view of these many distinctions, and for at least these reasons, claim 26 as amended is not anticipated by *Baker*. Therefore, the rejection of claim 26 as amended under 35 U.S.C. § 102(b) is overcome. Applicants respectfully submit that claim 26 as amended is allowable.

Claims 27-28 depend from claim 26 as amended and are allowable for the same reasons. Each of claims 27-28 has been amended to better accord with the language of claim 26 as amended.

Claim 29 has been added. Claim 29 is directed to a method of communicating over a target medium having a multi-layered protocol. The method recites, *inter alia*, the steps of —

> providing a software program;
>
> defining a first plurality of communication element types, accessible by the software program and representing different communication elements for a first layer of the protocol; and
>
> defining a second plurality of communication element types, accessible by the software program and representing different communication elements for a second layer of the protocol, the second layer being lower than the first layer.

The specification provides examples of the first and second plurality of communication element types. For instance, one may define a plurality of message types (see ¶36) and a plurality of word types (see ¶33). One may define a plurality of field types (see ¶32). Field types are lower than word types, and word types are lower than message types (see ¶¶34-37).

Baker does not disclose a step of "defining a first plurality of communication element types […] representing different communication elements for a first layer of the protocol." Neither does *Baker* disclose a step of "defining a second plurality of communication element types […] representing different communication elements for a second layer of the protocol." *Baker*'s programmably configurable protocol descriptions contain field subrecords, but these

relate to fields only. No structures are presented that relate to any other protocol layers, such as a message layer or a word layer.

Claim 29 also recites, *inter alia*, a step of —

> including, within the definition of at least one of the first plurality of communication element types, a reference to at least one of the second plurality of communication element types".

The specification provides a supporting example of this recitation at ¶37, wherein a message type is defined to include references to different word types, and at ¶34, wherein a word type is defined to include references to different field types.

Baker does not disclose this "including" step. Since *Baker's* programmably configurable protocol descriptions are organized around only one layer of the protocol (fields), *Baker* does not disclose a step that involves two layers of the protocol, i.e., the step of "including, within the definition of each of the first plurality of communication element types, a reference to at least one of the second plurality of communication element types" as recited in claim 29.

Claim 29 further recites, *inter alia*, a step of –

> instantiating one of the first plurality of communication element types to create a communication element instance thereof within the software program, the communication element instance including an instance of each of the second plurality of communication element types referenced by said one of the first plurality of communication element types, each instance being a specific expression of the respective communication element type.

The specification provides an example of this "instantiating" step at ¶49.

Baker does not disclose this "instantiating" step. As discussed above, *Baker* does not disclose separate "types" and "instances." Therefore, *Baker* does not disclose the creation of hierarchical instances from hierarchical types, as recited in the "instantiating" step of claim 29.

Claim 29 still further recites a step of "manipulating at least one of the communication element instances within the software program." *Baker* does not include this limitation. Even if *Baker*'s programmably configurable protocol descriptions can be modified or replaced for working with different protocols, *Baker* makes no mention of modifying them for "communicating over a target medium having a multi-layered protocol," as recited in claim 29. *Baker* does not create instances or manipulate them for communicating over a single protocol.

For at least these reasons, claim 29 distinguishes over *Baker*. Applicants contend that claim 29 is allowable.

Claims 30-34 have also been added. Each of these claims depends from claim 29 and is allowable for the same reasons.

Claim 35 has been added. Claim 35 is an apparatus claim directed to a computer-implemented system for communicating over a target medium having a multi-layered protocol. The system includes "a software program for controlling an electronic instrument connected to the target medium" and "a bus model file." The bus model file is accessible by the software program and includes:

> a first plurality of communication element types representing different communication elements for a first layer of the protocol; and
>
> a second plurality of communication element types representing different communication elements for a second layer of the protocol, the second layer being lower than the first layer;
>
> each of the first and second plurality of communication element types being an instantiable software data type, and
>
> at least one of the first plurality of communication element types including a reference to at least one of the second plurality of communication element types;

Baker does not disclose "a first plurality of communication element types representing different communication elements for a first layer of the protocol" and "a second plurality of communication element types representing different communication elements for a second layer of the protocol." *Baker*'s programmably configurable protocol descriptions contain field subrecords, but these relate to fields only. No structures are presented that relate to any other protocol layers, such as a message layer or a word layer.

Nothing in *Baker*'s programmably configurable protocol descriptions corresponds to "an instantiable software data type," as recited in claim 35. For reasons already discussed, *Baker* does not disclose the use of *data types* for representing protocols; *Baker* merely discloses the use of data.

Claim 35 further recites,

> a software API (applications program interface), accessible by the software program, for creating communication element instances based on the first and second plurality of communication element types, said communication element instances including—
>
> > at least one first communication element instance within the software program, each being a specific expression of a respective one of the first plurality of communication element types; and
> >
> > at least one second communication element instance within the software program, each being a specific expression of one of the second plurality of communication element types included by reference in a respective first communication element type.

Baker does not disclose a software API, as recited in claim 35. In addition, since *Baker* is concerned with only one layer of the protocol (fields), *Baker* does not disclose "at least one of

the first plurality of communication element types including a reference to at least one of the second plurality of communication element types," as required by claim 35.

As discussed in connection with other claims, *Baker* makes no distinction between "types" and "instances" when it comes to representing a protocol. Therefore, *Baker* does not disclose any "first communication element instance" or "second communication element instance," as recited in claim 35.

Furthermore, *Baker* does not disclose "a software program for controlling an electronic instrument connected to the target medium" or "a software API […] accessible by the software program, for creating communication element instances, as required by claim 35.

In view of these differences, claim 35 is not anticipated by *Baker*. Applicants respectfully submit that claim 35 is allowable.

Claims 36-38 have also been added. Each of claims 36-38 depends from claim 35 and is allowable for the same reasons.

Conclusion:

Applicants contend that this application is now in condition for allowance. A notice to that effect is earnestly solicited. If, after considering these amendments, the Examiner does not believe the case to be in condition for allowance, the Examiner is kindly requested to contact Applicants' attorney by telephone at the number provided below.

Respectfully Submitted,

/Bruce D. Rubenstein #39,349/

Bruce D. Rubenstein
Reg. No. 39,349
Attorney for Applicant

Atty. Docket : TERAD-8-US
Telephone : 781-274-0202
Fax : 781-274-0201